Electronic Notes in Theoretical Computer Science 70 No. 3 (2002)
URL: http://www.elsevier.nl/locate/entcs/volume70.html 16 pages

Taking Decisions Late:
End-Based Choice Combined with Action Refinement

Harald Fecher and Mila Majster-Cederbaum

Universität Mannheim
Fakultät für Mathematik und Informatik
D7, 27, 68131 Mannheim, Germany
{hfecher,mcb}@pi2.informatik.uni-mannheim.de

Abstract

A choice in concurrent systems is usually taken by the starting of an action. We propose the alternative view that a choice is determined by the ending of actions as this alternative has relevant applications and interesting implications. The different points of view lead in particular to different refinement operators.

We introduce a refinement operator on bundle event structures for the end-based view. The standard equivalences are not preserved by this refinement operator. Therefore, we also introduce and study new equivalences that are preserved by our refinement operator.

Key words: action refinement, event structure, true concurrency, equivalence, congruence

1 Introduction

Concurrent systems can be modeled, for example, by action based process algebras [19,25], by true concurrent or interleaving semantics models. In these approaches actions are usually considered to be instantaneous, i.e. durationless. If however real time aspects of systems have to be modeled and/or action refinement operators [16] are employed, we have to take into account that actions consume time.

If durational actions are considered, the following question arises: when does an action trigger a choice – at the beginning, at the end or anywhere in the middle of its duration?

The consequence of this decision is illustrated by the following example. Consider a process that has to make a choice ($+$) between actions a and b where the duration of a is 3 and the duration of b is 1. Furthermore, action a starts at time 0 and action b starts at time 1 if possible. If the choice is triggered at the beginning, then a triggers the choice, i.e. before b starts. On the other hand, if the choice is triggered by the end of an action, the choice is triggered by b, i.e. a does not finish.

The standard approach in the literature is to trigger the choice by the starting of an involved event (action) [3,17,24,26,30]. But it is reasonable to consider approaches where the choices are caused by the ending of actions. For example, in stochastic approaches it is common to consider a *race policy* approach [4,7,18], i.e. the fastest action triggers the choice. Consequently, a choice has to be triggered at the end of the action's duration, since it is usually not known a priori which action is the fastest.

The end-based point of view is also of interest for hierachical system development [16], where complex activities are specified by single actions in the first system design steps. This can be seen by the following example.

Example 1.1 Consider the example of a plane that runs into problems and has to land as fast as possible. Two airports (in the same city) come into consideration for the emergency landing. The pilot sends an SOS-signal to both airports. Both airports start their preparations for the emergency landing. The pilot will choose the airport that is the first one which responds to be ready. On an abstract level the pilot can be modeled by

$$P_{ab} = send; ((ok_1; L_1) + (ok_2; L_2))$$

where $send$ denotes the sending of the SOS-Signal, ok_i is the respond of the i-th airport A_1 (that runs in parallel to P_{ab} synchronizing over ok_i) and L_i denotes the landing on the i-th airport. The choice in P_{ab} is either triggered by ok_1 or by ok_2, as usual.

In practice, the airports will send more detailed information, e.g. that the maneuvering area is free, fire service is ready and so on. In other words, actions ok_1 and ok_2 are time consuming. Then the choice in P_{ab} has to be considered as end-based, since the choice should be made when the first airport has completed its preparations.

This is easily understood when we consider the next system design phase, where the ok_i actions are specified in more detail, i.e. are refined by a process M_i. Then the choice of the pilot is triggered when either M_1 or M_2 terminates and not by the first action performed by M_1 or M_2. In particular, the actions of M_2 that are performed before the termination of M_1 remain visible, i.e. are not made undone, after the termination of M_1 and vice versa. This makes clear that the end-based choice can be viewed as some kind of parallelism, where M_1 and M_2 run in parallel until one terminates. In [20] another kind of a parallel choice is presented in the context of an atomizer operator.

The end-based choice is an operator that models 'speculative' concurrency, which is practically important because speculation is almost the only way to reduce unpredictable latencies. Furthermore, such kind of operators can be used to evade the normal sequentiality restriction of most programming languages.

The possibility of late decisions is also motivated and examined in Z [27].

The decision when a choice is triggered does not influence the theory of most untimed semantic models. This situation changes when models that contain an

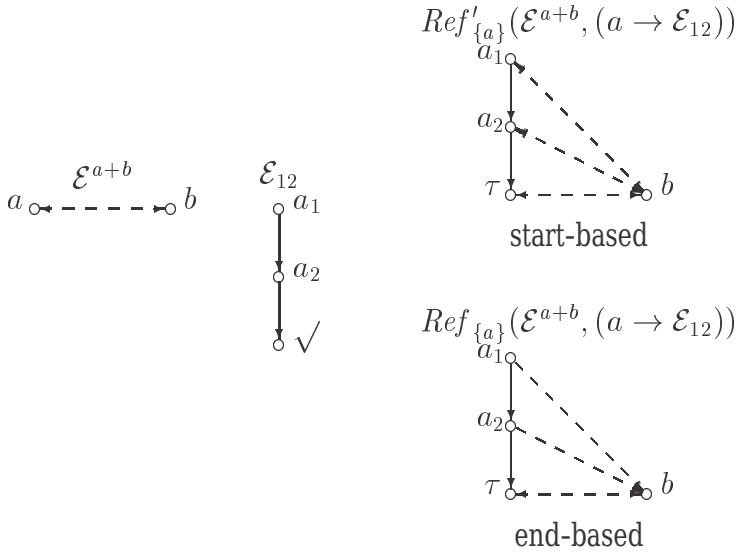


Fig. 1.

action refinement operator (see Example 1.1) are considered, which is an important paradigm for system development [14,16]. Action refinement can, for example, split an action into a start- and an end-action. More precisely, if action a is refined to a_1 followed by a_2, i.e. a_2 performs after a_1, in the process consisting of a choice between actions a and b, then the execution sequence a_1, b is allowed in an end-based setting, which is not the case in a start-based setting. In Figure 1 we illustrate this example by using event stuctures with an asymmetric conflict relation, where $e' \circ\!\!- - \!\!\to\!\circ e$ means that the performance of e disables the performance of e' but not vice versa. We depict the event structures after the refinement in the start-based and in the end-based approach.

In the standard approach to action refinement a choice is triggered by the starting of actions, for example as in [2,10,14,16,24,28].

In this paper we develop a theory of an end-based point of view in an untimed model that includes an action refinement operator. The underlying model is chosen to be *extended bundle event structures* [21,22]. These event structures are suitable, since each event in a bundle event structures represents a unique occurrence of an action, which is usually not the case for prime event structures, as it is pointed out, for example, in [14]. This unique representation is essential for the definition of our refinement operator.

It turns out that the action refinement operator in an end-based setting is not compatible with conventional equivalence notions. In most equivalences the processes $P_1 = a$ and $P_2 = a + a$ are identified. Refining a in an end-based setting leads usually to processes that are no longer equivalent under these equivalences. Therefore, new equivalences that are congruences with respect to this refinement operator have to be established. Three of them are presented: the ICT-equivalence, which is based on traces, and the UI- and FUI-equivalences, which are based on bisimulation.

The paper is organized as follows. Section 2 introduces closed bundle event

structures, which are a variant of extended bundle event structures mentioned before. The definition of our refinement operator is presented in Section 3. Section 4 is devoted to the equivalence notions. A conclusion is given in Section 5.

2 Closed Bundle Event Structures

Here we introduce the class of *closed bundle event structures* [11], which is a subclass of the class of *extended bundle event structures* [21]. This subset, which is determined by an additional closedness condition, is needed to give denotations to recursive processes, since extended bundle event structures fail to yield a complete partial order [1], which can be seen in [11].

We write $\mathcal{P}(M)$ to denote the powerset of M and we write $M_1 \rightharpoonup M_2$ to denote the set of all partial functions from M_1 to M_2. Furthermore, the domain of a partial function f is the set $\{m \mid f(m) \text{ is defined}\}$, and is denoted by $\mathrm{dom}(f)$. For any binary relation $\odot$ we write $p \odot q$ if and only if $(p, q) \in \odot$.

Let $\surd$ and τ be two different elements, which indicate the *terminating* and the *internal action*. Furthermore, let Obs be a set such that $\surd, \tau \notin$ Obs. We call Obs the set of *observable actions*. The set of all actions $\mathcal{A}ct$ is defined by $\mathcal{A}ct = \{\surd, \tau\} \cup$ Obs.

Definition 2.1 Let E be a countable set. A *finite, monotone approximation* of E is a sequence $(E_i)_{i\in\mathbb{N}}$ such that $\bigcup_{i\in\mathbb{N}} E_i = E \wedge \forall k : E_k \subseteq E_{k+1} \wedge |E_k| < \infty$.

Definition 2.2 [Closed bundle event structure] Suppose $\bullet$, $\star_1$, $\star_2$ and $\star$ are pairwise different. Assume a fixed countable set of events $\mathcal{U}$ such that $\bullet \in \mathcal{U}$, $\star \notin \mathcal{U}$ and $\forall e, e' \in \mathcal{U} : (e, e'), (\star_1, e), (\star_2, e), (\star, e), (e, \star) \in \mathcal{U}$.
A *closed bundle event structure* (c.b.e.s.) $\mathcal{E} = (E, \rightsquigarrow, \mapsto, l)$ is an element of $\mathcal{P}(\mathcal{U}) \times \mathcal{P}(\mathcal{U} \times \mathcal{U}) \times \mathcal{P}(\mathcal{P}(\mathcal{U}) \times \mathcal{U}) \times (\mathcal{U} \rightharpoonup \mathcal{A}ct)$ such that

- $\rightsquigarrow \subseteq (E \times E) \backslash \{(e, e) \mid e \in E\}$
- $\mapsto \subseteq \mathcal{P}(E) \times E$
- $\mathrm{dom}(l) = E$
- $\forall X \subseteq E, e \in E : X \mapsto e \Rightarrow (\forall e', e'' \in X : e' \neq e'' \Rightarrow e' \rightsquigarrow e'')$
- $X \mapsto e$ whenever $X \subseteq E$ and there is a finite, monotone approximation $(E_n)_{n\in\mathbb{N}}$ of E such that $\forall n : \exists X' : X' \mapsto e \wedge X' \cap E_n = X \cap E_n$

Let $\mathbf{CEBES}$ denote the set of all closed bundle event structures.

We call E the set of events, $\rightsquigarrow$ the (irreflexive) *asymmetric conflict* relation, $\mapsto$ the *bundle* relation and l the *action-labeling* function of $\mathcal{E}$. Hereafter, we consider $\mathcal{E}$ to be $(E, \rightsquigarrow, \mapsto, l)$, $\mathcal{E}_i$ to be $(E_i, \rightsquigarrow_i, \mapsto_i, l_i)$ and in general $\mathcal{E}$ to be $(E_\mathcal{E}, \rightsquigarrow_\mathcal{E}, \mapsto_\mathcal{E}, l_\mathcal{E})$.

The intuitive meaning of a c.b.e.s. is the following: If e is in conflict to e', i.e. $e \rightsquigarrow e'$, then e' disables e forever, but not vice versa. $X \mapsto e$ means that before e may be performed an event from X has to be performed. The condition on the bundle X guarantee that exactly one event of X has to be performed before e. The

labeling function indicates which action is observable when the event is performed. The presented closedness condition guarantees that $\mathbf{CEBES}$ yields a complete partial order [11].

Some useful notations are presented in the following definition. $\mathrm{init}(\mathcal{E})$ denotes all events which are ready to perform, $\mathrm{init}_A(\mathcal{E})$ denotes all events which are ready to perform and have labels in $A \subseteq \mathcal{A}ct$ and $\mathrm{exit}(\mathcal{E})$ denotes those events which correspond to the termination.

Definition 2.3 The set of *initial events* of the c.b.e.s. $\mathcal{E}$ is defined by $\mathrm{init}(\mathcal{E}) = \{e \in E \mid \neg(\exists X : X \mapsto e)\}$. Furthermore, define $\mathrm{init}_A(\mathcal{E}) = \{e \in \mathrm{init}(\mathcal{E}) \mid l(e) \in A\}$. The set of *successful termination events* of the c.b.e.s. $\mathcal{E}$ is defined by $\mathrm{exit}(\mathcal{E}) = \{e \in E \mid l(e) = \surd\}$.

For the sake of simplicity we write $\mathrm{init}_a(\mathcal{E})$ instead of $\mathrm{init}_{\{a\}}(\mathcal{E})$. We define the remainder [5,22,23] of a c.b.e.s. to illustrate how a c.b.e.s. behaves after the performance of an event. The remainder is used to define an interleaving semantics for $\mathbf{CEBES}$. We denote the restriction of the function f to the set M by $f^{\restriction}M$.

Definition 2.4 [Remainder of a c.b.e.s.] Let $\mathcal{E} \in \mathbf{CEBES}$ and $e \in \mathrm{init}(\mathcal{E})$. Then the remainder $\mathcal{E}_{[e]}$ of $\mathcal{E}$ is given by $(E', \rightsquigarrow', \mapsto', l')$ where

$$
\begin{aligned}
E' &= \{e' \in E \mid e' \neq e \wedge \neg(e' \rightsquigarrow e)\} \\
\rightsquigarrow' &= \rightsquigarrow \cap (E' \times E') \\
\mapsto' &= \{(X \cap E', e') \mid e' \in E' \wedge X \mapsto e' \wedge e \notin X\} \\
l' &= l^{\restriction}E'
\end{aligned}
$$

Then a transition system can be obtained as follows.

Definition 2.5 The transition relation $\longrightarrow \subseteq \mathbf{CEBES} \times \mathcal{A}ct \times \mathbf{CEBES}$ is defined by $- \xrightarrow{} \{(\mathcal{E}, l(e), \mathcal{E}_{[e]}) \mid \mathcal{E} \in \mathbf{CEBES} \wedge e \in \mathrm{init}(\mathcal{E})\}$.

3 Operators on CEBES

3.1 Standard Operators

The definition for the standard operators (as choice, sequential and parallel composition) on $\mathbf{CEBES}$ can be given in the usual way [11,21,22] and is hence omitted here.

3.2 Refinement Operator

In the following we present our definition of a refinement operator. It differs from the classical definition with respect to the conflict relation: only the terminating events of the refining processes are used in our approach to define the conflict relation whereas every event (or every initial event) is used in the standard approach. More precisely, in the classical definition we have: if e is in conflict with e' and e

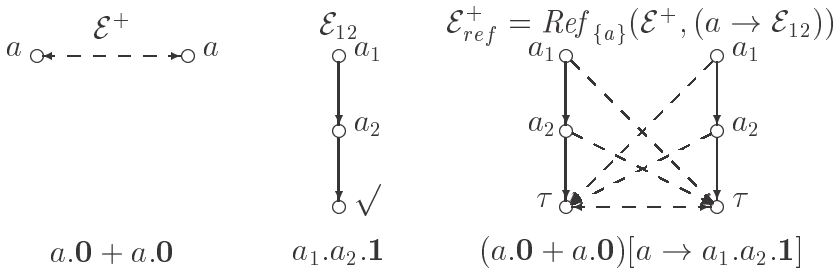


Fig. 2.

(e' respectively) gets refined to $\mathcal{E}_e$ ($\mathcal{E}_{e'}$ respectively), then every event of $\mathcal{E}_e$ is put in conflict with every event of $\mathcal{E}_{e'}$ and vice versa. In our definition, we put every event of $\mathcal{E}_e$ in conflict with the terminating events of $\mathcal{E}_{e'}$, i.e. they may only perform if they perform before every terminating event of $\mathcal{E}_{e'}$. And of course, we put every event of $\mathcal{E}_{e'}$ in conflict with the terminating events of $\mathcal{E}_e$. By this approach we guarantee that a choice is triggered by the ending of actions, i.e. by a terminating event of the refining process.

Definition 3.1 [Refinement Operator] Let $A \subseteq$ Obs. Define $Ref_A : \mathbf{CEBES} \times (A \to \mathbf{CEBES}) \to \mathbf{CEBES}$ by $Ref_A(\mathcal{E}, \theta) = (\tilde{E}, \tilde{\rightsquigarrow}, \tilde{\mapsto}, \tilde{l})$ where

$$\begin{aligned}
\tilde{E} &= \{(e, e') \mid e \in E \wedge l(e) \in A \wedge e' \in E_{\theta(l(e))}\} \cup \\
&\quad \{(e, e) \in E \times E \mid l(e) \notin A\} \\
\tilde{\rightsquigarrow} &= \{((e_1, e_1'), (e_2, e_2')) \mid (e_1 \rightsquigarrow e_2 \wedge (l(e_2) \in A \Rightarrow e_2' \in \mathrm{exit}(\theta(l(e_2))))) \vee \\
&\quad (e_1 = e_2 \wedge l(e_1) \in A \wedge e_1' \neq e_2' \wedge (e_1' \rightsquigarrow_{\theta(l(e_1))} e_2' \vee e_2' \in \mathrm{exit}(\theta(l(e_1)))))\} \\
\tilde{\mapsto} &= \{(\{e\} \times X', (e, e')) \mid l(e) \in A \wedge X' \mapsto_{\theta(l(e))} e'\} \cup \\
&\quad \{(\tilde{X}, (e, e')) \mid \exists X : X \mapsto e \wedge (l(e) \in A \Rightarrow e' \in \mathrm{init}(\theta(l(e)))) \wedge \\
&\quad \tilde{X} = \{(\hat{e}, \hat{e}') \in \tilde{E} \mid \hat{e} \in X \wedge (l(\hat{e}) \in A \Rightarrow \hat{e}' \in \mathrm{exit}(\theta(l(\hat{e}))))\}\}
\end{aligned}$$

$$\tilde{l}((e, e')) = \begin{cases} l(e) & \text{if } l(e) \notin A \\ l_{\theta(l(e))}(e') & \text{if } l(e) \in A \wedge l_{\theta(l(e))}(e') \neq \surd \\ \tau & \text{if } l(e) \in A \wedge l_{\theta(l(e))}(e') = \surd \end{cases}$$

Example 3.2 We illustrate by a small example in Figure 2 how the refinement operator (Ref) behaves. For a better understanding, we augment the examples by process term descriptions of the systems, where $[a \to a_1.a_2.\mathbf{1}]$ denotes the refinement operator on the process terms and $(a \to \mathcal{E}_{12})$ denotes the function from $\{a\}$ to $\mathbf{CEBES}$ that maps a to $\mathcal{E}_{12}$.

The example in Figure 2 illustrates that the events labeled by a_1 are not in conflict in $\mathcal{E}^+_{ref}$. Only the events that correspond to the termination of $\mathcal{E}_{12}$, which are labeled by τ in $\mathcal{E}^+_{ref}$, disable the other actions. In other words, both actions a in $\mathcal{E}^+_{ref}$ may start and perform their actions independently until one of them terminates.

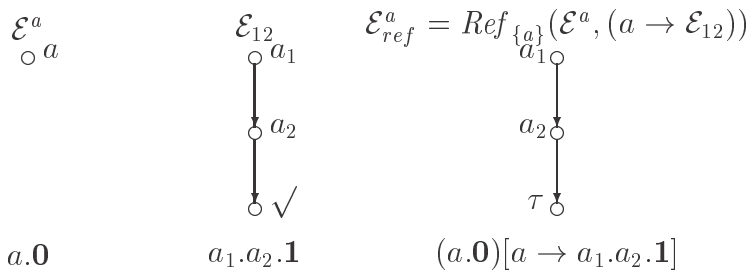


Fig. 3.

Our refinement operator allows the modeling of an interrupt mechanism as it is used, for example, in LOTOS [8]. This is illustrated as follows. Suppose the process described by $\mathcal{E}_2$ can interrupt the process described by $\mathcal{E}_1$, denoted by $\mathcal{E}_1[> \mathcal{E}_2$. Then let $\mathcal{E}^a + \mathcal{E}_2$ be the process that is obtained by taking a choice between $\mathcal{E}_2$ and the event structure $\mathcal{E}^a$ from Figure 3 where label a does not appear in $\mathcal{E}_2$. Then the process $\mathcal{E}_1[> \mathcal{E}_2$ is obtained by $Ref_{\{a\}}(\mathcal{E}^a + \mathcal{E}_2, (a \to \mathcal{E}_1))$.

4 Equivalences

Event structures used as system description are too concrete, i.e. processes with the same behavior may be described by different event structures. Therefore, equivalences are defined on event structures, and on other models, to identify those event structures with the same behavior. The basic equivalences (observable behaviors) are the trace equivalence and the bisimulation equivalence.

Definition 4.1 [Trace Equivalence] The set of traces of $\mathcal{E} \in \mathbf{CEBES}$ is defined by $T(\mathcal{E}) = \{\emptyset\} \cup \{(l(e_i))_{i \leq n} \mid n \in \mathbb{N} \wedge \exists \mathcal{E}_0, \cdots \mathcal{E}_{n+1} : \mathcal{E}_0 = \mathcal{E} \wedge \forall i \leq n : \mathcal{E}_{i[e_i]} = \mathcal{E}_{i+1}\}$.

Two c.b.e.s. $\mathcal{E}$ and $\mathcal{E}'$ are *trace equivalent*, denoted $\mathcal{E} \sim_{tr} \mathcal{E}'$, if $T(\mathcal{E}) = T(\mathcal{E}')$.

Definition 4.2 [Strong Bisimilarity] Two c.b.e.s. $\mathcal{E}_1$ and $\mathcal{E}_2$ are *strong bisimilar*, denoted $\mathcal{E}_1 \sim_{sb} \mathcal{E}_2$, if there is a symmetric relation $\mathcal{R} \subseteq \mathbf{CEBES} \times \mathbf{CEBES}$ such that $(\mathcal{E}_1, \mathcal{E}_2) \in \mathcal{R}$ and for all $(\mathcal{E}_1', \mathcal{E}_2') \in \mathcal{R}$ we have:

if $\mathcal{E}_1' \xrightarrow{a} \mathcal{E}_1''$ then there is $\mathcal{E}_2''$ such that $(\mathcal{E}_1'', \mathcal{E}_2'') \in \mathcal{R}$ and $\mathcal{E}_2' \xrightarrow{a} \mathcal{E}_2''$.

Trace equivalence and bisimulation are not preserved under the usual refinement operator [14], whereas ST-equivalence was shown to be the coarsest congruence for classical action refinement [12,15,29]. In the case of our refinement operator any equivalence that implies trace equivalence and identifies a and $a+a$ is not preserved, since $\mathcal{E}^a_{ref}$ from Figure 3 and $\mathcal{E}^+_{ref}$ from Figure 2 do not even have the same traces. Consequently, the standard equivalences, like trace, bisimulation, step, pomset, history preserving equivalences etc. [13,14], are not preserved. Resource bisimulation [9], which is defined on transition systems, is the only equivalence known to us that does not identify $a + a$ and a.

In the following subsections we present new equivalences which are indeed congruences with respect to our refinement operator.

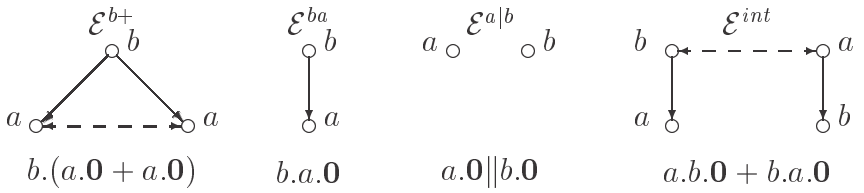


Fig. 4.

4.1 ICT-Equivalence

The first equivalence notion we consider is based on trace equivalence. An equivalence notion which is a congruence for our refinement operator has to distinguish between $\mathcal{E}^+$ and $\mathcal{E}^a$. One way to achieve this is to guarantee that the number of the initial events which are labeled by the same action have to be equal, i.e. $\mathcal{E}$ and $\mathcal{E}'$ can only be equivalent if $|\text{init}_a(\mathcal{E})| = |\text{init}_a(\mathcal{E})|$ for all $a \in \mathsf{Obs}$. Moreover, we also have to guarantee a relationship between the numbers of the initial events with the same label of the corresponding remainders of the event structures. For example, consider $\mathcal{E}^{b+}$ and $\mathcal{E}^{ba}$ from Figure 4. Then $(b, a_1, a_1) \in T(Ref_{\{a\}}(\mathcal{E}^{b+}, (a \to \mathcal{E}_{12})))$ but $(b, a_1, a_1) \notin T(Ref_{\{a\}}(\mathcal{E}^{ba}, (a \to \mathcal{E}_{12})))$.

Further difficulties arise as can be seen by considering $\mathcal{E}^{a|b}$ and $\mathcal{E}^{int}$ from Figure 4: $\mathcal{E}^{a|b}$ and $\mathcal{E}^{int}$ satisfy our above criterion, but $(a_1, b, a_1) \in T(Ref_{\{a\}}(\mathcal{E}^{int}, (a \to \mathcal{E}_{12})))$ and $(a_1, b, a_1) \notin T(Ref_{\{a\}}(\mathcal{E}^{a|b}, (a \to \mathcal{E}_{12})))$, i.e. our tentative relation is not a congruence with respect to the refinement.

Therefore, we introduce the *initial event traces* of a c.b.e.s. They consist of an event execution sequence and of a subset of the initial events for every execution step. Two c.b.e.s. $\mathcal{E}_1$ and $\mathcal{E}_2$ are considered to be equivalent if every initial event trace of $\mathcal{E}_1$ can be mapped by an injective function f to an initial event trace of $\mathcal{E}_2$ and vice versa. Furthermore, this function has to be labeling-preserving, i.e. $\forall e_1 \in E_1 : l_1(e_1) = l_2(f(e_1))$. The equivalence is precisely defined in the following definition, where $\mathcal{P}_{fin}(M)$ denotes the set of all finite subsets of M, i.e. $\mathcal{P}_{fin}(M) = \{M' \subseteq M \mid |M'| < \infty\}$.

Definition 4.3 [ICT-equivalence] Let $\mathcal{E} \in \mathbf{CEBES}$. Then the *initial event traces* of $\mathcal{E}$ are defined by $T^{ic}(\mathcal{E}) = \{(e_i, \gamma_i)_{i \le n} \mid n \in \mathbb{N} \wedge \exists \mathcal{E}_0, \cdots, \mathcal{E}_{n+1} : \mathcal{E}_0 = \mathcal{E} \wedge \forall i \le n : \mathcal{E}_{i[e_i]} = \mathcal{E}_{i+1} \wedge \gamma_i \in \mathcal{P}_{fin}(\text{init}_{\mathsf{Obs}}(\mathcal{E}_i))\}$.
Two c.b.e.s. $\mathcal{E}$ and $\mathcal{E}'$ are *initial corresponding trace equivalent* (ICT-equivalent), denoted $\mathcal{E} \sim_{ICT} \mathcal{E}'$, if

- for every $(e_i, \gamma_i)_{i \le n} \in T^{ic}(\mathcal{E})$ there is an injective, labeling-preserving function $f : (\bigcup_{i \le n}(\gamma_i \cup \{e_i\})) \to E'$ such that $(f(e_i), f(\gamma_i))_{i \le n} \in T^{ic}(\mathcal{E}')$ [1]
- for every $(e'_i, \gamma'_i)_{i \le n} \in T^{ic}(\mathcal{E}')$ there is an injective, labeling-preserving function $f' : (\bigcup_{i \le n}(\gamma'_i \cup \{e'_i\})) \to E$ such that $(f'(e'_i), f'(\gamma'_i))_{i \le n} \in T^{ic}(\mathcal{E})$

$\mathcal{E}^{b+}$ and $\mathcal{E}^{ba}$ and also $\mathcal{E}^{a|b}$ and $\mathcal{E}^{int}$ from Figure 4 are not ICT-equivalent. In addition the event structures from Figure 5 are also not ICT-equivalent. This holds,

[1] $f(\gamma_i)$ denotes the image of the set γ_i under f

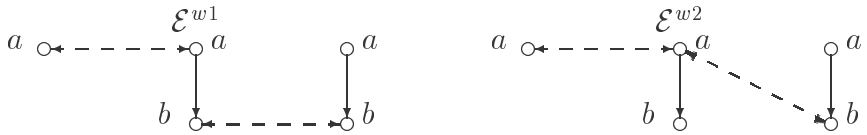
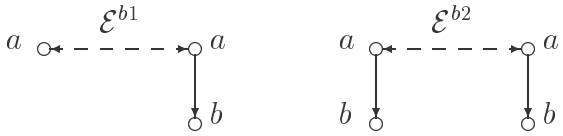
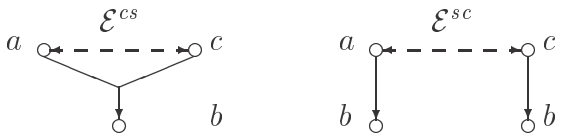


Fig. 5.



$a.\mathbf{0} + a.b.\mathbf{0}$ $\quad$ $a.b.\mathbf{0} + a.b.\mathbf{0}$

Fig. 6.



$(a + c).b.\mathbf{0}$ $\quad$ $a.b.\mathbf{0} + c.b.\mathbf{0}$

Fig. 7.

since in $\mathcal{E}^{w1}$ it is possible that both events labeled by b become enabled, which is not the case for $\mathcal{E}^{w2}$. Examples of ICT-equivalent event structures are given in Figure 6 and in Figure 7.

Proposition 4.4 *ICT-equivalent c.b.e.s. are also Trace-equivalent, i.e.* $\sim_{ICT} \subset \sim_{tr}$.

Theorem 4.5 *The ICT-equivalence is a congruence for the refinement operator, i.e.* $\mathcal{E} \sim_{ICT} \mathcal{E}' \wedge \forall a \in A : \theta(a) \sim_{ICT} \theta'(a)$ *implies that* $Ref_A(\mathcal{E}, \theta) \sim_{ICT} Ref_A(\mathcal{E}', \theta')$. *It is also preserved under the standard operators on* $\mathbf{CEBES}$*, like choice, sequential composition and parallel composition.*

Remark 4.6 Strong bisimilarity does not follow from ICT-equivalence. This is the case, since the c.b.e.s. $\mathcal{E}^{b1}$ and $\mathcal{E}^{b2}$ from Figure 6 are not bisimilar but ICT-equivalent.

4.2 UI-Bisimilarity

An equivalence that is based on bisimulation equivalence and that is a congruence for our refinement operator has to relate the initial events as it is also done in the ICT-equivalence. Therefore, we extend the bisimulation by a third component which denotes a bijection between the initial events.

Definition 4.7 [UI-Bisimulation] A *unique initial (UI) bisimulation* $\mathcal{R}$ is a subset of $\mathbf{CEBES} \times \mathbf{CEBES} \times (\mathcal{U} \rightharpoonup \mathcal{U})$ such that whenever $(\mathcal{E}_1, \mathcal{E}_2, f) \in \mathcal{R}$ then

- $\mathrm{dom}(f) = \mathrm{init}_{\mathsf{Obs}}(\mathcal{E}_1)$

- f is an labeling-preserving bijection between $\text{init}_{\mathsf{Obs}}(\mathcal{E}_1)$ and $\text{init}_{\mathsf{Obs}}(\mathcal{E}_2)$
- $e_1 \in \text{init}(\mathcal{E}_1)$ implies that there is e_2 and f' such that $l_1(e_1) = l_2(e_2)$ and $(\mathcal{E}_{1[e_1]}, \mathcal{E}_{2[e_2]}, f') \in \mathcal{R}$ and $f \cup f'$ [2] is an injective function and $l_1(e_1) \in \mathsf{Obs} \Rightarrow e_2 = f(e_1)$
- and $(\mathcal{E}_2, \mathcal{E}_1, f^{-1}) \in \mathcal{R}$

We say that $\mathcal{E}_1, \mathcal{E}_2$ are *UI-bisimilar* (or *UI-equivalent*), denoted $\mathcal{E}_1 \sim_{UI} \mathcal{E}_2$, if and only if there is a UI-bisimulation $\mathcal{R}$ and an $f \in \mathcal{P}(\mathcal{U} \times \mathcal{U})$ such that $(\mathcal{E}_1, \mathcal{E}_2, f) \in \mathcal{R}$.

The event structures from Figure 6 are not UI-equivalent, whereas the event structures from Figure 7 are UI-equivalent.

The condition that $f \cup f'$ has to be a function in Definition 4.7 is used to guarantee that the identification of the initial events of $\mathcal{E}_1$ is preserved after the execution, i.e. $f^{\restriction}(\text{init}_{\mathsf{Obs}}(\mathcal{E}_1) \cap \text{init}_{\mathsf{Obs}}(\mathcal{E}_{1[e_1]})) = f'^{\restriction}(\text{init}_{\mathsf{Obs}}(\mathcal{E}_1) \cap \text{init}_{\mathsf{Obs}}(\mathcal{E}_{1[e_1]}))$. The condition $f \cup f'$ is an *injective* function guarantees that an initial action e_1' of $\mathcal{E}_1$ is kept after the execution if and only if $f(e_1')$ is kept after the corresponding execution, i.e. $e_1' \in \text{init}_{\mathsf{Obs}}(\mathcal{E}_1) \cap \text{init}_{\mathsf{Obs}}(\mathcal{E}_{1[e_1]}) \Leftrightarrow f(e_1') \in \text{init}_{\mathsf{Obs}}(\mathcal{E}_2) \cap \text{init}_{\mathsf{Obs}}(\mathcal{E}_{2[e_2]})$. This means that the identification of the initial events of $\mathcal{E}_2$ is preserved after the execution, i.e. $f^{-1\restriction}(\text{init}_{\mathsf{Obs}}(\mathcal{E}_2) \cap \text{init}_{\mathsf{Obs}}(\mathcal{E}_{2[e_2]})) = f'^{-1\restriction}(\text{init}_{\mathsf{Obs}}(\mathcal{E}_2) \cap \text{init}_{\mathsf{Obs}}(\mathcal{E}_{2[e_2]}))$.

Proposition 4.8 *If two c.b.e.s. are UI-equivalent then they are also strong bisimilar, i.e.* $\sim_{UI} \subset \sim_{sb}$.

Theorem 4.9 *The UI-equivalence is a congruence for the refinement operator, i.e.* $\mathcal{E} \sim_{UI} \mathcal{E}' \wedge \forall a \in A : \theta(a) \sim_{UI} \theta'(a)$ *implies that* $Ref_A(\mathcal{E}, \theta) \sim_{UI} Ref_A(\mathcal{E}', \theta')$. *It is also preserved under the standard operators on* $\mathbf{CEBES}$*, like choice, sequential composition and parallel composition.*

4.3 FUI-Bisimilarity

The UI-equivalence has to preserve all initial events correspondence. This condition is not necessary to obtain a congruence that is contained in strong bisimilarity. It is sufficient to guarantee that the correspondence of any finite subset of the initial events is preserved. This is formalized in the following definition.

Definition 4.10 [FUI-Bisimulation] A *finite unique initial (FUI) bisimulation* $\mathcal{R}$ is a subset of $\mathbf{CEBES} \times \mathbf{CEBES} \times (\mathcal{U} \rightharpoonup \mathcal{U})$ such that whenever $(\mathcal{E}_1, \mathcal{E}_2, f) \in \mathcal{R}$ then

- $\text{dom}(f) = \text{init}_{\mathsf{Obs}}(\mathcal{E}_1)$
- f is an labeling-preserving bijection between $\text{init}_{\mathsf{Obs}}(\mathcal{E}_1)$ and $\text{init}_{\mathsf{Obs}}(\mathcal{E}_2)$
- $e_1 \in \text{init}(\mathcal{E}_1) \wedge I \in \mathcal{P}_{fin}(\text{init}_{\mathsf{Obs}}(\mathcal{E}_1))$ implies that there is e_2 and f' such that $l_1(e_1) = l_2(e_2)$ and $(\mathcal{E}_{1[e_1]}, \mathcal{E}_{2[e_2]}, f') \in \mathcal{R}$ and $l_1(e_1) \in \mathsf{Obs} \Rightarrow e_2 = f(e_1)$ and $f^{\restriction}(I \cap \text{init}_{\mathsf{Obs}}(\mathcal{E}_{1[e_1]})) = f'^{\restriction} I$ and $f^{-1\restriction}(f(I) \cap \text{init}_{\mathsf{Obs}}(\mathcal{E}_{2[e_2]})) = f'^{-1\restriction} f(I)$
- and $(\mathcal{E}_2, \mathcal{E}_1, f^{-1}) \in \mathcal{R}$

[2] $f \cup f'$ is defined by considering f and f' as relations

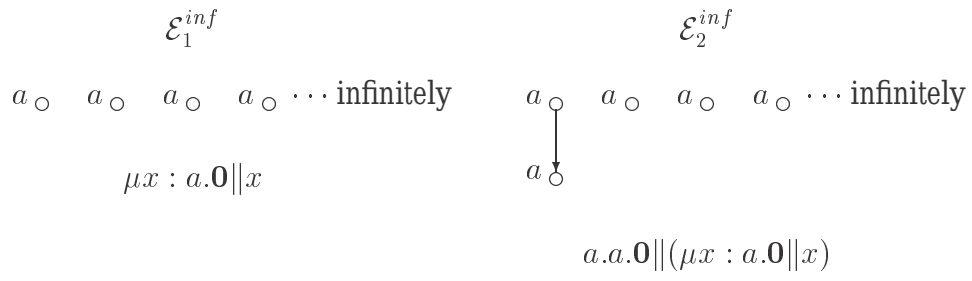


Fig. 8.

We say that $\mathcal{E}_1, \mathcal{E}_2$ are *FUI-bisimilar* (or *FUI-equivalent*), denoted $\mathcal{E}_1 \sim_{FUI} \mathcal{E}_2$, if and only if there is a FUI-bisimulation $\mathcal{R}$ and an $f \in \mathcal{P}(\mathcal{U} \times \mathcal{U})$ such that $(\mathcal{E}_1, \mathcal{E}_2, f) \in \mathcal{R}$.

Remark 4.11 Obviously, any c.b.e.s. with a finite set of events is UI-equivalent if and only if it is FUI-equivalent.

The UI-equivalence is different to the FUI-equivalence, since the event structures from Figure 8 are FUI-equivalent but not UI-equivalent.

Proposition 4.12 *If two c.b.e.s. are FUI-equivalent then they are also strong bisimilar, i.e.* $\sim_{FUI} \subset \sim_{sb}$.

Theorem 4.13 *The FUI-equivalence is a congruence for the refinement operator, i.e.* $\mathcal{E} \sim_{FUI} \mathcal{E}' \wedge \forall a \in A : \theta(a) \sim_{FUI} \theta'(a)$ *implies that* $Ref_A(\mathcal{E}, \theta) \sim_{FUI} Ref_A(\mathcal{E}', \theta')$. *It is also preserved under the standard operators on* **CEBES**, *like choice, sequential composition and parallel composition.*

4.4 Relations between the Equivalences

First, we present the connection between ICT-equivalence, UI-equivalence and FUI-equivalence.

Proposition 4.14 *UI-equivalent c.b.e.s. are also FUI-equivalent. Furthermore, FUI-equivalent c.b.e.s. are also ICT-equivalent, i.e.*

$$\sim_{UI} \subset \sim_{FUI} \subset \sim_{ICT} .$$

From Proposition 4.8 and Remark 4.6, we obtain that the second inclusion in Proposition 4.14 is strict. The connection between the equivalences is summarized in Figure 9. That ICT-equivalence is not comparable with strong bisimilarity follows from Remark 4.6 and from the fact that a and $a + a$ are bisimilar but not ICT-equivalent. Since all equivalence notions from [14] identify a and $a + a$, they cannot be contained in $\sim_{ICT}$ and consequently cannot be contained in $\sim_{UI}$ and in $\sim_{UI}$.

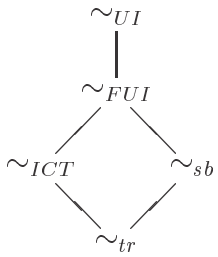


Fig. 9. Relations between the equivalences

4.5 Coarsest Congruence

In this subsection we define the coarsest equivalence for strong bisimulation with respect to the refinement operator Ref. It turns out that it is different from the FUI-equivalence, i.e. the FUI-equivalence is not the coarsest equivalence for strong bisimulation. Furthermore, the ICT-equivalence also fails to be the coarsest equivalence for the trace equivalence.

Definition 4.15 Define $\sim_c \subseteq \mathbf{CEBES} \times \mathbf{CEBES}$ by $\mathcal{E}_1 \sim_c \mathcal{E}_2$ if and only if $\forall A \subseteq \mathsf{Obs} : \forall \theta : A \to \mathbf{CEBES} : Ref_A(\mathcal{E}_1, \theta) \sim_{sb} Ref_A(\mathcal{E}_2, \theta)$.

Proposition 4.16 *The relation $\sim_c$ is the coarsest congruence for $\sim_{sb}$ with respect to the refinement operator Ref.*

The FUI-equivalence is a proper subset of $\sim_c$, i.e. the FUI-equivalence is not the coarsest congruence with respect to Ref. This is illustrated by the following example.

Example 4.17 Consider $\mathcal{E}^{c1}$ and $\mathcal{E}^{c2}$ from Figure 10. They are not ICT-equivalent and therefore, also not FUI-equivalent, since after performing action a the number of the initial actions does not coincide. But $\mathcal{E}^{c1} \sim_c \mathcal{E}^{c2}$, which can be seen as follows. If a is not refined then both c.b.e.s., which are bisimilar, remain unchanged under the refinement. Now suppose that a is refined. Then we do not have a problem to find a corresponding bisimilar process for every execution step as long as the refinement of a does not terminate. When the refinement terminates, then the process performs τ. If '$\mathcal{E}^{c1}$' performs this τ then '$\mathcal{E}^{c2}$' can perform its shown to the left τ to yield a bisimilar c.b.e.s. If '$\mathcal{E}^{c2}$' performs this τ then '$\mathcal{E}^{c1}$' can perform its shown to the right τ to yield a bisimilar c.b.e.s.

Also the ICT-equivalence is not the coarsest congruence for $\sim_{tr}$ with respect to Ref, since $\sim_c$ is a congruence with respect to Ref and $\sim_c \subseteq \sim_{sb} \subseteq \sim_{tr}$ but $\sim_c \not\subseteq \sim_{ICT}$ by Example 4.17.

The reason why $\sim_{ICT}$ and $\sim_{FUI}$ fail to be coarsest congruences for Ref stems from the fact that Ref renames events labeled by $\surd$ to τ. This renaming is necessary for the well-definedness of this operator, since a definition that removes these events will not result in an element of $\mathbf{CEBES}$ or will not respect the intuitive meaning.

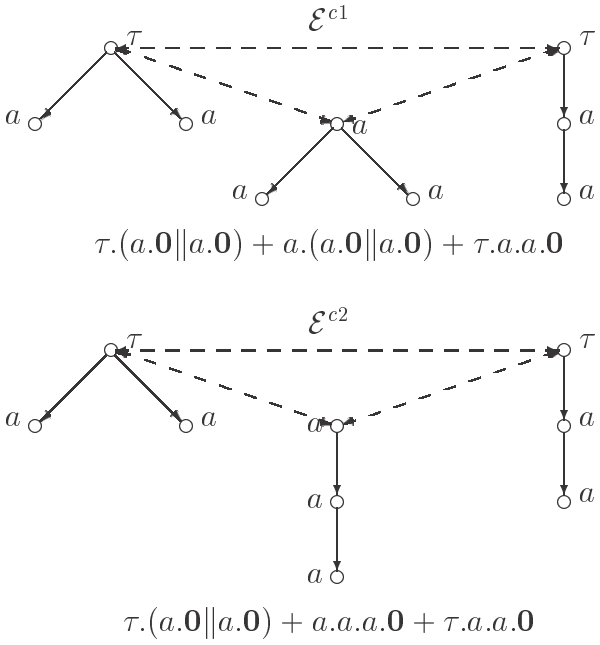


Fig. 10. Counterexample for coarsest congruence

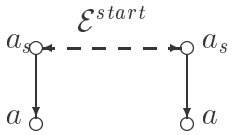


Fig. 11.

5 Conclusion

We have presented a refinement operator on closed bundle event structures. It considers choices as end-based triggered, i.e. a choice is triggered by the ending of an action and not by the starting. Furthermore, ICT-, UI- and FUI-equivalence have been introduced; they are congruence relations with respect to our refinement operator and the standard operators on event structures. The newly introduced equivalences do not yield coarsest equivalences. This fact depends on the considered event structures. The relation between ICT-, UI-, FUI-, trace-equivalence and strong bisimilarity has been studied.

The modeling of a start-based choice in our setting can be derived by explicitly adding an instantaneous action for the start of an action. More precisely, if in the c.b.e.s. $\mathcal{E}^+$ from Figure 2 the choice has to be taken by the start of the actions, we consider instead the c.b.e.s. $\mathcal{E}^{start}$ from Figure 11. Action a_s in $\mathcal{E}^{start}$ presents the start of action a, and it has to be instantaneous, i.e. it may not be refined. The disadvantage of this approach is that we add actions which stay observable throughout the whole top-down design.

A better approach to model a start-based choice is obtained if the event structures are extended by another conflict relation that corresponds to the start-based choices. This conflict relation behaves in the classical way under action refinement [14].

Future work is to investigate a process algebra that allows the possibility to undo the activities of the rejected alternative in an end-based choice setting. This would yield a formal method that can model 'speculative' concurrency. The main problem in doing so is to handle communication that has already taken place in the rejected alternative.

References

[1] Samson Abramsky and Achim Jung. Domain theory. In Samson Abramsky, Dov M. Gabbay, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 3, pages 1–168. Clarendon Press, 1994.

[2] L. Aceto and M. Hennessy. Adding action refinement to a finite process algebra. *Information and Computation*, 115:179–247, 1994.

[3] Luca Aceto and David Murphy. Timing and causality in process algebra. *Acta Informatica*, 33:317–350, 1996.

[4] Marco Ajmone Marsan, Gianni Conte, and Gianfranco Balbo. A class of generalized stochastic petri nets for the performance evaluation of multiprocessor systems. *ACM Transactions on Computer Systems*, 2:93–122, 1984.

[5] Christel Baier and Mila E. Majster-Cederbaum. The connection between an event structure semantics and an operational semantics for TCSP. *Acta Informatica*, 31:81–104, 1994.

[6] J. A. Bergstra, A. Ponse, and S. A. Smolka, editors. *Handbook of Process Algebra*. North-Holland, 2001.

[7] Marco Bernardo and Roberto Gorrieri. A tutorial on EMPA: A theory of concurrent processes with nondeterminism, priorities, probabilities and time. *Theoretical Computer Science*, 202:1–54, 1998.

[8] Tommaso Bolognesi and Ed Brinksma. Introduction to the ISO specification language LOTOS. *Computer Networks and ISDN Systems*, 14:25–59, 1987.

[9] Flavio Corradini, Rocco De Nicola, and Anna Labella. Graded modalities and resource bisimulation. In C. Pandu Rangan, V. Raman, and R. Ramanujam, editors, *FSTTCS'99*, volume 1738 of *Lecture Notes in Computer Science*, pages 381–393. Springer-Verlag, 1999.

[10] Philippe Darondeau and Pierpaolo Degano. Refinement of actions in event structures and causal trees. *Theoretical Computer Science*, 118:21–48, 1993.

[11] Harald Fecher, Mila Majster-Cederbaum, and Jinzhao Wu. Bundle event structures: A revised cpo approach. *Information Processing Letters*, 83:7–12, 2002.

[12] R.J. van Glabbeek. The refinement theorem for ST-bisimulation semantics. In M. Broy and C.B. Jones, editors, Proceedings IFIP TC2 Working Conference on *Programming Concepts and Methods,* Sea of Gallilee, Israel, April 1990, pages 27–52. North Holland, 1990.

[13] R.J. van Glabbeek. The linear time–branching time spectrum I. the semantics of concrete, sequential processes. In Bergstra et al. [6], pages 3–99.

[14] Rob van Glabbeek and Ursula Goltz. Refinement of actions and equivalence notions for concurrent systems. *Acta Informatica*, 37:229–327, 2001.

[15] Roberto Gorrieri and Cosimo Laneve. Split and ST bisimulation semantics. *Information and Computation*, 118:272–288, 1995.

[16] Roberto Gorrieri and Arend Rensink. Action refinement. In Bergstra et al. [6], pages 1047–1147.

[17] Roberto Gorrieri, Marco Roccetti, and Enrico Stancampiano. A theory of processes with durational actions. *Theoretical Computer Science*, 140:73–94, 1995.

[18] Holger Hermanns and Michael Rettelbach. Syntax, semantics, equivalences, and axioms for mtipp. In U. Herzog and M. Rettelbach, editors, *Proceedings of the 2nd Int. Workshop on Process Algebra and Performance Modelling (PAPM'94)*, 1994.

[19] C. A. R. Hoare. *Communications Sequential Processes*. International Series in Computer Science. Prentice Hall, 1985.

[20] Peter M. W. Knijnenburg and Joost N. Kok. The semantics of the combination of atomized statements and parallel choice. *Formal Aspects of Computing*, 9:518–536, 1997.

[21] Rom Langerak. *Transformations and Semantics for LOTOS*. PhD thesis, Department of Computer Science, University of Twente, 1992.

[22] Rom Langerak. Bundle event structures: A non-interleaving semantics for LOTOS. In M. Diaz and R. Groz, editors, *Formal Description Techniques, V*, pages 331–346. Elsevier, 1993.

[23] Mila Majster-Cederbaum and Markus Roggenbach. Transition systems from event structures revisited. *Information Processing Letters*, 67:119–124, 1998.

[24] Mila Majster-Cederbaum and Jinzhao Wu. Action refinement for true concurrent real-time. In *Proc. 7th IEEE int. Conf. on Engineering of Complex Computer Systems*, pages 58–68. IEEE Computer Sciety Press, 2001.

[25] Robin Milner. *Communication and Concurrency*. International Series in Computer Science. Prentice Hall, 1989.

[26] David Murphy. Time and duration in noninterleaving concurrency. *Fundamenta Informaticae*, 19:403–416, 1993.

[27] Susan Stepney, David Cooper, and Jim Woodcock. More powerful data refinement in Z: pushing the state of the art in industrial refinement. In J.P. Bowen, A. Fett, and M.G. Hinchey, editors, *ZUM'98: The Z Formal Specification Notation*, volume 1493 of *Lecture Notes in Computer Science*, pages 284–307. Springer-Verlag, 1998.

[28] Walter Vogler. Failures semantics based on interval semiwords is a congruence for refinement. *Distributed Computing*, 4:139–162, 1991.

[29] Walter Vogler. Bisimulation and action refinement. *Theoretical Computer Science*, 114:173–200, 1993.

[30] Walter Vogler. Timed testing of concurrent systems. *Information and Computation*, 121:149–171, 1995.